日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００４年　３月　３日

出願番号
Application Number: 特願２００４－０５９８６５

[ST. 10/C]: [ＪＰ２００４－０５９８６５]

出願人
Applicant(s): オムロン株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

２００４年　３月２４日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

【書類名】　特許願
【整理番号】　62947
【提出日】　平成16年 3月 3日
【あて先】　特許庁長官殿
【国際特許分類】　G05B 19/05
【発明者】
　【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内
　【氏名】　岡　実
【発明者】
　【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内
　【氏名】　出来　仁太郎
【発明者】
　【住所又は居所】　京都府京都市下京区西洞院木津屋橋通東入ル　オムロンソフトウエア株式会社内
　【氏名】　矢尾板　宏心
【発明者】
　【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内
　【氏名】　市村　勝彦
【発明者】
　【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内
　【氏名】　小野　彰男
【特許出願人】
　【識別番号】　000002945
　【氏名又は名称】　オムロン株式会社
　【代表者】　作田　久男
【代理人】
　【識別番号】　100092598
　【弁理士】
　【氏名又は名称】　松井　伸一
【先の出願に基づく優先権主張】
　【出願番号】　特願2003- 68920
　【出願日】　平成15年 3月13日
【手数料の表示】
　【予納台帳番号】　019068
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9800459

【書類名】特許請求の範囲
【請求項１】
　Ｉ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するＣＰＵユニットを含むプログラマブルコントローラと、そのプログラマブルコントローラに接続され、前記ユーザプログラムを編集するためのプログラミングツールと、からなる制御装置であって、
　前記プログラミングツールは、
　プログラマブルコントローラに対してユーザプログラムをアップロードおよびダウンロードする手段を持ち、
　前記プログラマブルコントローラのＣＰＵユニットは、
　ユーザプログラム格納用の２つのメモリと、
サイクリック処理動作中に前記プログラミングツールからユーザプログラム編集処理がなされる場合に、ユーザプログラム実行処理の実行対象をいずれか一方のメモリに格納されているユーザプログラムと定める手段と、
　前記プログラミングツールのアップロードによって、サイクリック処理動作の周辺処理期間中に、プログラミングツールに対して一方または他方のメモリのいずれかに格納されているユーザプログラムを出力する手段と、
　前記プログラミングツールからダウンロードされてきたユーザプログラムを、サイクリック処理動作の周辺処理期間中に、ユーザプログラム実行処理の実行対象としていない前記他方のメモリへ格納する手段と、
　その格納が完了したことを条件に、ユーザプログラム実行処理の実行対象を前記一方のメモリから前記他方のメモリに切り替える手段と、
前記他方のメモリに格納されたユーザプログラムを前記一方のメモリへコピー格納する手段と、
を持つことを特徴とする制御装置。
【請求項２】
　請求項１記載の制御装置において、
サイクリック処理動作中に前記プログラミングツールからユーザプログラム編集処理がなされる場合に、ユーザプログラム実行処理の実行対象を、他方のメモリに格納されているユーザプログラムから一方のメモリに格納されているユーザプログラムへ切り替える手段、
をさらに持つことを特徴とする制御装置。
【請求項３】
　プログラミングツールとの間でユーザプログラムをアップロードおよびダウンロードでき、Ｉ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するＣＰＵユニットであって、
　ユーザプログラム格納用の２つのメモリと、
サイクリック処理動作中に前記プログラミングツールからユーザプログラム編集処理がなされる場合に、ユーザプログラム実行処理の実行対象をいずれか一方のメモリに格納されているユーザプログラムと定める手段と、
　前記プログラミングツールのアップロードにより、サイクリック処理動作の周辺処理期間中に、前記プログラミングツールに対して一方または他方のメモリのいずれかに格納されているユーザプログラムを出力する手段と、
　前記プログラミングツールからダウンロードされてきたユーザプログラムを、サイクリック処理動作の周辺処理期間中に、ユーザプログラム実行処理の実行対象としていない前記他方のメモリへ格納する手段と、
　その格納が完了したことを条件に、ユーザプログラム実行処理の実行対象を前記一方のメモリから前記他方のメモリに切り替える手段と、
前記他方のメモリに格納されたユーザプログラムを前記一方のメモリへコピー格納する手段と、

を持つことを特徴とするＣＰＵユニット。
【請求項４】
ユーザプログラムを格納する２つのメモリを持ってＩ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するＣＰＵユニットが接続されたプログラマブルコントローラに対して、プログラミングツールから前記ユーザプログラムを編集する方法であって、
ＣＰＵユニットにて予め、前記２つのメモリに格納しているユーザプログラムを同一の内容としておき、
ＣＰＵユニットにて、プログラミングツールからユーザプログラムを編集処理するのに先立ち、ユーザプログラムの演算実行対象を、いずれか一方のメモリに格納されているユーザプログラムと定めておき、ＣＰＵユニットのサイクリック処理動作をそのまま継続させ、
プログラミングツールにて、ＣＰＵユニットの動作中に、ＣＰＵユニットの一方または他方のメモリのいずれかに格納されているユーザプログラムをアップロードし、そのユーザプログラムを編集し、
プログラミングツールにて、その編集を終了した後、ＣＰＵユニットが動作中に、前記プログラミングツールから前記他方のメモリへ編集後のユーザプログラムをダウンロードして格納し、
ＣＰＵユニットにて、その格納を終了した後、前記ユーザプログラムの演算実行対象を前記一方のメモリから前記他方のメモリに切り替えて、ＣＰＵユニットが編集後のユーザプログラムを実行処理するようにし、
ＣＰＵユニットにて、前記他方のメモリに格納された編集後のユーザプログラムを前記一方のメモリに格納して、両メモリのユーザプログラムを同一内容とするようにした
ことを特徴とするプログラマブルコントローラのユーザプログラム編集方法。
【請求項５】
請求項４記載のプログラマブルコントローラのユーザプログラム編集方法であって、
前記プログラミングツールにおいてユーザプログラムをアップロードおよびダウンロードする際には、その対象メモリを、ＣＰＵユニットの動作中におけるユーザプログラム演算実行対象のほうのメモリでないもう一方のメモリとする
ことを特徴とするプログラマブルコントローラのユーザプログラム編集方法。
【請求項６】
請求項４記載のプログラマブルコントローラのユーザプログラム編集方法であって、
前記プログラミングツールにおいてユーザプログラムをアップロードおよびダウンロードする際には、ＣＰＵユニットの動作中における周辺処理期間中に行うことを特徴とするプログラマブルコントローラのユーザプログラム編集方法。
【請求項７】
請求項４に記載のプログラマブルコントローラのユーザプログラム編集方法であって、
新しく次回、プログラミングツールにてユーザプログラムを編集する際、
ＣＰＵユニットにて、ユーザプログラムの演算実行対象を前記他方のメモリから前記一方のメモリに切り替え、その状態を編集後のユーザプログラムのダウンロードが終了するまで維持する
ことを特徴とするプログラマブルコントローラのユーザプログラム編集方法。
【請求項８】
請求項４に記載のプログラマブルコントローラのユーザプログラム編集方法であって、
新しく次回、プログラミングツールにてユーザプログラムを編集する際、
ＣＰＵユニットにて、ユーザプログラムの演算実行対象を前記他方のメモリのままでＣＰＵユニットの動作を継続させ、
プログラミングツールにて、ユーザプログラムの編集を終了した後、ＣＰＵユニットがサイクリック処理動作中に、前記一方のメモリへ編集後のユーザプログラムをダウンロードして格納し、

ＣＰＵユニットにて、その格納を終了した後、前記ユーザプログラムの演算実行対象を前記他方のメモリから前記一方のメモリに切り替えて、ＣＰＵユニットが新しく編集したユーザプログラムを実行処理するようにした
ことを特徴とするプログラマブルコントローラのユーザプログラム編集方法。
【請求項９】
ユーザプログラムを格納する２つのメモリを持ち、Ｉ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するプログラマブルコントローラにおいて、プログラミングツールによって前記ユーザプログラムをオンラインエディットされる際のプログラマブルコントローラの処理方法であって、
予め、前記２つのメモリに格納しているユーザプログラムをそれぞれ同一の内容とし、ユーザプログラムの演算実行対象を２つのメモリのいずれか一方のメモリに格納されているユーザプログラムと定めるとともに、プログラミングツールからの書き込み可能対象を他方のメモリのみと定め、
プログラマブルコントローラのサイクリック処理動作をそのまま継続させながら、
プログラマブルコントローラの周辺処理動作中に、前記プログラミングツールに対してどちらかのメモリに格納されているユーザプログラムを送信し、
プログラマブルコントローラの周辺処理動作中に、前記プログラミングツールからのユーザプログラムを前記他方のメモリへ格納し、
ユーザプログラムの演算実行対象を前記一方のメモリから前記他方のメモリに切り替えて、前記他方のメモリに格納されたユーザプログラムを実行処理するようにし、
プログラマブルコントローラの周辺処理動作中に、前記他方のメモリに格納されたユーザプログラムを前記一方のメモリへ格納して、２つのメモリのユーザプログラムを同一内容とするようにした
ことを特徴とするオンラインエディットされる際のプログラマブルコントローラの処理方法。

【書類名】明細書
【発明の名称】制御装置、ＣＰＵユニット、プログラマブルコントローラのユーザプログラム編集方法、及びオンラインエディットされる際のプログラマブルコントローラの処理方法
【技術分野】
【０００１】
この発明は、ユーザメモリを演算実行するＣＰＵユニットやプログラマブルコントローラにおけるユーザプログラム編集に関するものである。
【背景技術】
【０００２】
ファクトリーオートメーション（ＦＡ）の制御装置として、プログラマブルコントローラ（ＰＬＣ）が用いられている。このＰＬＣは、複数のユニットから構成される。すなわち、電源供給源の電源ユニット，ＰＬＣ全体の制御を統率するＣＰＵユニット，ＦＡの生産装置や設備装置の適所に取り付けたスイッチやセンサの信号を入力する入力ユニット，アクチュエータなどに制御出力を出す出力ユニット，通信ネットワークに対する処理、例えばプログラミングツールや上位モニタ装置、他のＰＬＣ等に対してデータ通信する処理を行う通信ユニットなどの各種のユニットを適宜組み合わせて構成される。
【０００３】
ＰＬＣのＣＰＵユニットにおける制御は、まず初期処理をし、その後、入力ユニットで入力した信号をＣＰＵユニットのＩ／Ｏメモリに取り込み（ＩＮリフレッシュ処理）、ユーザプログラム記述言語（例えばラダー言語）により組まれ、メモリに予め登録されたユーザプログラムに基づき論理演算をし（演算実行処理）、その演算実行結果をＩ／Ｏメモリに書き込んで出力ユニットに送り出し（ＯＵＴリフレッシュ処理）、その後、いわゆる周辺処理を行う。そして、ＩＮリフレッシュ処理と演算実行処理とＯＵＴリフレッシュ処理と周辺処理とをサイクリックに繰り返し処理するようになる。なお初期処理の後のサイクリック処理の順は別の変形例でもよく、例えば、ＯＵＴリフレッシュ処理とＩＮリフレッシュ処理（まとめてＩ／Ｏリフレッシュ処理とも言う）、演算実行処理、周辺処理でもよい。この場合、最初のＯＵＴリフレッシュ処理は初期処理による固定値を出力し、次サイクルのＯＵＴリフレッシュ処理から演算実行結果を出力することとなる。また、サイクリック処理の順を、演算実行処理、Ｉ／Ｏリフレッシュ処理、周辺処理とする変形例でもよい。この場合、最初の演算実行処理は初期処理による固定値の演算を行い、次サイクルの演算実行処理からＩＮリフレッシュ処理で取り込んだＩＮデータに基づく演算を行うこととなる。
【０００４】
ユーザプログラムは、ＰＬＣに対してネットワーク接続されたプログラミングツールによって作成または編集される。作成後または編集後のユーザプログラムは、プログラミングツールからＰＬＣへダウンロードされ、ＣＰＵユニットのユーザメモリに格納される。このユーザプログラムは、いわゆるラダー言語で構成されたラダープログラムによって複数の入力接点の論理回路が組まれている。なお、ユーザプログラムには、ひとかたまりの動作をプログラミングしたファンクションブロックを構成要素に含んでいる場合もある。ファンクションブロックは、複数の演算回路を組み合わせてひとつに部品化されたものである。
【０００５】
ところで、実際にＰＬＣのＣＰＵユニットにてユーザプログラムを実行（サイクリック処理の演算実行）し、生産設備や設備装置を稼動した後においても、プログラム上に不具合を発見したり、動作の不具合が発生したりすることがある。その場合には、ユーザプログラムに対して追加・削除・変更などの編集を行う必要が生じる。係るプログラムの編集処理の一態様として、ＰＬＣの制御を実行しつつ、プログラムの編集を行うオンラインエディットがある。
【０００６】

従来、このオンラインエディットを行う場合には、プログラミングツールによってユーザプログラム全体の中の編集対象のプログラム部分の直前にジャンプ命令等を一時的に追加するようにしていた。そして、ＰＬＣがユーザプログラムを演算実行する際に、編集対象部分だけを飛ばしてユーザプログラムを実行することにより、ＰＬＣを停止することなく動作を継続できていた。そしてツールにて、ＰＬＣが動作中の状態で、ユーザプログラムの編集対象部分に対し追加・削除・変更などの編集作業を行うようにしていた。なお、上記の発明と類似する先行技術としては、特許文献１がある。

【０００７】

【特許文献１】特開２００１－１４２５１０号公報

【発明の開示】

【発明が解決しようとする課題】

【０００８】

しかしながら、上記した従来のオンラインエディットは、１回の処理で１つの回路に対して行うことは比較的簡単にできても、ユーザプログラムの複数箇所に点在するラダー回路を一度に修正するのは困難であった。その理由は、ユーザプログラム中の複数の編集箇所すべてにジャンプ命令を一時的に追加するのは大変面倒であったからである。また編集後に、一時的に追加したジャンプ命令等をすべて削除する必要があり面倒であった。さらにジャンプ命令をひとつでも削除し忘れた場合には、編集したプログラムが正常に動作しなくなり、不具合が生じていた。

【０００９】

また、ファンクションブロックに対するオンラインエディットも従来実行できなかった。すなわち、ファンクションブロックはユーザプログラム中の一箇所にまとまって格納されることはまれで、複数箇所に分散していることが多い。これは、ファンクションブロックの構成要素が、メインのラダープログラムに対してサブルーチンプログラムのようになっていることと、インスタンスとボディとからなっていることから、通常はそれらがプログラム中の複数箇所に分散して格納されるからである。よって、従来の通常の回路に対する編集の方法をファンクションブロックに適用すると、前述と同様に、ジャンプ命令を挿入する作業が多く発生してしまい、その作業が大変面倒であった。またジャンプ命令を削除するのも面倒で、そのジャンプ命令をひとつでも削除し忘れると編集したファンクションブロックプログラムが正常に動作しないこととなり、不具合であった。

【００１０】

ファンクションブロックの事情をもう少し詳しく述べる。ユーザプログラムを格納するユーザプログラムメモリ全領域は、ファンクションブロックのボディ部が複数格納されるファンクションブロックレコード部（記録領域）と、実際のメインのラダープログラム部分とファンクションブロックが挿入されている場合にはそれに対応するファンクションブロックの入力・出力を特定するインスタンスとが格納されるプログラムレコード部（記憶領域）とに区分けされている（詳しくは図８でも述べる）。よってファンクションブロックに対して編集処理を行うということは、入力と出力のインスタンス部分と、実際のファンクションブロックのボディ部分との複数箇所を同時に編集することとなる。よって、上述のメインユーザプログラムのラダー回路の複数箇所を一度に修正する場合と同じく、ジャンプ命令挿入および削除作業が多く発生してしまうこととなる。

【００１１】

オンラインエディットのもうひとつの不具合を言うと、ユーザプログラム中の編集対象部分はジャンプ命令によって処理がスキップされるので、制御に必要不可欠な動作に対応するプログラム部分に対して編集処理すると、スキップ処理に伴ってＰＬＣ動作、ひいては生産設備の動作に悪影響を与えるおそれがある。従って、プログラムのどの部分に対しても自由にオンラインエディットが行えるとは限らない。

【００１２】

この発明は、ユーザプログラムの編集対象がどこであっても、また編集箇所が多くなっても（ファンクションブロック編集も含む）、オンラインエディットによる編集処理が行

えるＣＰＵユニット及びユーザプログラム編集方法を提供することを目的とする。
【課題を解決するための手段】
【００１３】
この発明による制御装置は、Ｉ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するＣＰＵユニット（１０）を含むプログラマブルコントローラと、そのプログラマブルコントローラに接続されてユーザプログラムを編集するためのプログラミングツール（１０）とからなり、プログラミングツールは、プログラマブルコントローラに対してユーザプログラムをアップロードおよびダウンロードする手段と、ユーザプログラムを編集する手段と、を持ち、プログラマブルコントローラのＣＰＵユニットは、ユーザプログラム格納用の２つのメモリ（ＵＭ１４，Ｔ－ＵＭ１９）と、サイクリック処理動作中にプログラミングツールからユーザプログラム編集処理がなされる場合に、ユーザプログラム実行処理の実行対象をいずれか一方のメモリに格納されているユーザプログラムと定める手段（ＭＰＵ１２）と、プログラミングツールのアップロードによって、サイクリック処理動作の周辺処理期間中にプログラミングツールに対して一方または他方のメモリのいずれかに格納されているユーザプログラムを出力する手段（ＭＰＵ１２，通信Ｉ／Ｆ１８）と、プログラミングツールからダウンロードされてきたユーザプログラムをサイクリック処理動作の周辺処理期間中にユーザプログラム実行処理の実行対象としていない他方のメモリへ格納する手段（ＭＰＵ１２）と、その格納が完了したことを条件にユーザプログラム実行処理の実行対象を一方のメモリから前記他方のメモリに切り替える手段（ＭＰＵ１２）と、他方のメモリに格納されたユーザプログラムを一方のメモリへコピー格納する手段（ＭＰＵ１２）とを持つ構成とした。
【００１４】
また、この発明によるＣＰＵユニット（１０）は、プログラミングツールとの間でユーザプログラムをアップロードおよびダウンロードでき、Ｉ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するもので、ユーザプログラム格納用の２つのメモリ（ＵＭ１４，Ｔ－ＵＭ１９）と、サイクリック処理動作中にプログラミングツールからユーザプログラム編集処理がなされる場合にユーザプログラム実行処理の実行対象をいずれか一方のメモリに格納されているユーザプログラムと定める手段（ＭＰＵ１２）と、プログラミングツールのアップロードによりサイクリック処理動作の周辺処理期間中にプログラミングツールに対して一方または他方のメモリのいずれかに格納されているユーザプログラムを出力する手段（ＭＰＵ１２，通信Ｉ／Ｆ１８、）と、プログラミングツールからダウンロードされてきたユーザプログラムをサイクリック処理動作の周辺処理期間中にユーザプログラム実行処理の実行対象としていない他方のメモリへ格納する手段（ＭＰＵ１２）と、その格納が完了したことを条件にユーザプログラム実行処理の実行対象を一方のメモリから他方のメモリに切り替える手段（ＭＰＵ１２）と、他方のメモリに格納されたユーザプログラムを一方のメモリへコピー格納する手段（ＭＰＵ１２）と、を持つ構成とした。
【００１５】
一方、本発明に係るユーザプログラム編集方法は、ユーザプログラムを格納する２つのメモリ（ＵＭ，Ｔ－ＵＭ）を持ってＩ／Ｏリフレッシュ処理とユーザプログラム実行処理と周辺処理とをサイクリックに処理するＣＰＵユニット（１０）が接続されたプログラマブルコントローラに対して、プログラミングツール（２０）からユーザプログラムを編集する方法であって、ＣＰＵユニットにて、予め前記２つのメモリに格納しているユーザプログラムを同一の内容としておき、プログラミングツールからユーザプログラムを編集処理するのに先立ち、ユーザプログラムの演算実行対象をいずれか一方のメモリに格納されているユーザプログラムと定めておき、ＣＰＵユニットのサイクリック処理動作をそのまま継続させ（図７（ａ）または図１０（ａ））、プログラミングツールにてＣＰＵユニットの動作中に、ＣＰＵユニットの一方または他方のメモリのいずれかに格納されているユーザプログラムをアップロードし（図４のＳＴ１１）、そのユーザプログラムを編集し（図４のＳＴ１２）、その編集を終了した後（図４のＳＴ１３）、他方のメモリへ編集後の

ユーザプログラムをダウンロードして格納し（図４のＳＴ１４、図６のＳＴ２３やＳＴ２４、図７のｂ、図１０のｂ）、ＣＰＵユニットにて、その格納を終了した後（図６のＳＴ２５）、ユーザプログラムの演算実行対象を一方のメモリから他方のメモリに切り替えてＣＰＵユニットが編集後のユーザプログラムを実行処理するようにし（図６のＳＴ２６、図７ｃ、図１０ｃ）、他方のメモリに格納された編集後のユーザプログラムを一方のメモリに格納して両メモリのユーザプログラムを同一内容とする（図６ＳＴ２７、図７ｄ、図１０ｅ）ようにした。

【００１６】

また、新しく次回、プログラミングツールにてユーザプログラムを編集する際、ＣＰＵユニットにて、ユーザプログラムの演算実行対象を他方のメモリから一方のメモリに切り替え、その状態を編集後のユーザプログラムのダウンロードが終了するまで維持するようにしてもよい。これは実施形態の図７ｄの状態から図７ａ、図７ｂへの処理流れに対応する。

【００１７】

また、新しく次回、プログラミングツールにてユーザプログラムを編集する際の変形例として、ＣＰＵユニットにて、ユーザプログラムの演算実行対象を他方のメモリのままでＣＰＵユニットの動作を継続させ、プログラミングツールにて、ユーザプログラムの編集を終了した後、ＣＰＵユニットがサイクリック処理動作中に一方のメモリへ編集後のユーザプログラムをダウンロードして格納し、ＣＰＵユニットにて、その格納を終了した後、ユーザプログラムの演算実行対象を他方のメモリから一方のメモリに切り替えて、ＣＰＵユニットが新しく編集したユーザプログラムを実行処理するようにすることでもよい。これは実施形態の図１０ｅの状態から図１０ｆ、それ以降の処理に該当する。

【００１８】

また、本発明に係るユーザプログラムをオンラインエディットする際のプログラマブルコントローラの処理方法は、予め、ＣＰＵユニットの２つのメモリに格納しているユーザプログラムをそれぞれ同一の内容とし、ユーザプログラムの演算実行対象を２つのメモリのいずれか一方のメモリに格納されているユーザプログラムと定めるとともに、プログラミングツールからの書き込み可能対象を他方のメモリのみと定め（図７ａ、図１０ａ）、ＣＰＵユニットのサイクリック処理動作をそのまま継続させながら、　ＣＰＵユニットの周辺処理動作中に、プログラミングツールに対して２つのメモリのいずれかに格納されているユーザプログラムを送信し、プログラミングツールから編集後のユーザプログラムを他方のメモリへ格納し（図６のＳＴ２３，２４、図７ｂ、図１０ｂ）、ユーザプログラムの演算実行対象を一方のメモリから他方のメモリに切り替えて編集後のユーザプログラムを実行処理するようにし（図６のＳＴ２６、図７ｃ、図１０ｃ）、ＣＰＵユニットの周辺処理動作中に他方のメモリに格納された編集後のユーザプログラムを一方のメモリへ格納して２つのメモリのユーザプログラムを同一内容とする（図６のＳＴ２７、図７ｄ、図１０ｅ）ようにした。

【００１９】

ＣＰＵユニットのユーザメモリに格納されたユーザプログラムに対してオンラインエディット編集をする場合、２つのユーザメモリには同一内容のユーザプログラムが格納されているので、一方のメモリのユーザプログラムを編集処理をしている間、他方のユーザメモリのユーザプログラムを演算実行に使うことができる。これにより、ＣＰＵユニットが演算実行するときに参照するユーザプログラムそのものに対しては編集処理をしないので、何ら制限無く編集処理と演算実行処理の両方が行える。

【００２０】

そして、オンラインエディット編集が完了したならば、今まで編集対象になっていたほうのメモリを演算実行対象に切替えることにより、編集された新しいユーザプログラムに基づく演算処理がすぐに行える。

【発明の効果】

【００２１】

以上のように、この発明では、ＣＰＵユニットにユーザメモリに加えてテンポラリユーザメモリを設け、その２つのメモリに同一のユーザプログラムを格納しておき、プログラミングツールが一方のユーザメモリに対してオンラインエディット編集処理をしている際に、ＣＰＵユニットが他方のテンポラリユーザメモリを参照してユーザプログラム実行処理を行う。オンラインエディット対象のユーザプログラムの格納場所とＣＰＵユニットにおけるユーザプログラム演算処理対象のユーザプログラムの格納場所とが独立しているので、自由にオンラインエディット処理ができるようになる。よって、従来のオンラインエディット処理時の不具合はなくなり、編集対象が制御に必須の部分であっても、また編集箇所が多くなっても（ファンクションブロック編集も含む）、オンラインエディットによる編集処理が行える。

【発明を実施するための最良の形態】

【００２２】

図１は、システム全体を示している。ＣＰＵユニット１０は、ＰＬＣの構成要素の一つである。そして、ＣＰＵユニット１０の本来のサイクリック処理動作をするためのシステムプログラムは、システムＲＯＭ１１に格納されている。そして、ＭＰＵ１２において、ワークメモリとしてのシステムＲＡＭ１３を適宜使用しながら、システムＲＯＭ１１のシステムプログラムに従った所定の処理を実行する。また、ＰＬＣの運転時において、ユーザがプログラミングツールを用いて予め作成したユーザプログラムは、ユーザメモリ１４に格納されている。さらに、ユーザプログラム実行の論理演算に用いるＩ／Ｏデータやパラメータは、ＩＯメモリ１５に格納されている。そしてＣＰＵユニット１０は、ユーザプログラム実行処理の前のＩＮリフレッシュ処理においてＩＮデータを、ＰＬＣシステムバス１７を介して接続されたＩ／Ｏユニット（図示なし）から取り込む。そしてＣＰＵユニット１０における論理演算処理ではそのＩＮデータに基づいて論理演算し、ＯＵＴリフレッシュ処理にてその論理演算結果をＯＵＴデータとしてＩ／Ｏユニットに対して出力する。

【００２３】

ＣＰＵユニット１０におけるユーザプログラム実行処理は、ＭＰＵ１２と命令実行エンジン１６とで行われる。ＭＰＵ１２は、ユーザプログラム実行処理に入ると、命令実行エンジン（ＡＳＩＣ）１６に対してユーザメモリ１４に格納されたユーザプログラム（命令オブジェクトコード）を順次呼び出して実行するように指示をする。その指示によって命令実行エンジン１６は、呼び出した命令オブジェクトを解析し、命令実行エンジン１６がもつ命令テーブルを参照し、プログラム命令実行をする。命令テーブルには、命令実行エンジン１６が自ら実行できる基本命令（ＡＮＤやＯＲの論理演算、簡単な数値演算など）が定義されている。命令テーブルにない応用命令（複雑な計算制御など）については、命令実行エンジン１６自らは実行できない。そこで、命令実行エンジン１６は、ＭＰＵ１２に対して、応用命令の実行依頼と、その実行に必要なＩＮデータやパラメータのデータ送信とを行う。このように命令実行エンジン１６とＭＰＵ１２とで命令実行の分担をするのは、基本命令を得意とするエンジンと応用命令を得意とするＭＰＵとによって処理の最大効率を図るためである。変形例として、ＭＰＵ１２の機能と命令実行エンジン１６の機能とをひとつのプロセッサで実現するような構成としてもよい。他の変形例として、命令実行エンジン１６が応用命令の一部についても実行できるようにしてもよい。（以下の説明では、ユーザプログラムは、命令実行エンジン１６が実行対処できるものであることとする）。

【００２４】

ユーザプログラム実行時に、命令実行エンジン１６は適宜ＩＯメモリ１５にアクセスし、ＩＮデータを読出したり、ＯＵＴデータを書き込んだり、またはパラメータを読出して取得する。通信インタフェース１８は、ツール２０との間でデータの送受を行うものである。この通信インタフェース１８を介してツール２０は、ＣＰＵユニット１０のＩ／Ｏデータをモニタしたり、ユーザメモリ１４に格納されているユーザプログラムをアップロードしたり、編集したユーザプログラムをユーザメモリ１４に対してダウンロードしたりす

る。このツール２０との間の通信処理は、ＣＰＵユニット１０におけるサイクリック処理の周辺処理にて行う。

【００２５】

本実施の形態のＰＬＣは、ファンクションブロック（ＦＢ）が実行できる。図２（ａ）はメインのユーザプログラムの一部であり、ファンクションブロックは図２（ａ）に示すようにラダープログラム中に挿入されている。このファンクションブロックは、引数とも呼ばれる入力パラメータ（図示の例では「Ｄ０，Ｄ１」）と、戻り値とも呼ばれる出力パラメータ（図示の例では「Ｄ２，Ｈ１．０」）と、実際に演算実行する処理を規定するボディＢとを備える。ボディＢの中身は、例えば図２（ｂ）に示すようになっていて、ひとかたまりの動作をするためのラダー言語でかかれたプログラムからなっている。メインのユーザプログラム中では、図２（ａ）に示すようにファンクションブロックのボディ部は箱で記述される。このボディ部の中身は、図２（ｂ）に示すような複数の回路からなるプログラムからなっている。実際のユーザプログラム演算実行に際してファンクションブロックのところまで来ると、ファンクションブロックのボディ部が呼び出され、入力パラメータに基づいて実行され、その実行結果が出力パラメータとして出力される。

【００２６】

ユーザプログラムの実行フローを図３に示す。ＣＰＵユニット１０の命令実行エンジン１６は、ユーザメモリＵＭ１４のユーザプログラムを上から順番に実行していく（ＳＴ１）。ユーザプログラム中に図２（ａ）のようにファンクションブロックが含まれていて、そのファンクションブロックを実行する順番になれば、命令実行エンジン１６は、ファンクションブロックの呼び出しをする（ＳＴ２）。これでユーザメモリのファンクションブロックレコード部の該当ＦＢ記憶領域にジャンプする。そして、命令実行エンジン１６は、インスタンス前処理に移る。インスタンス前処理では、ファンクションブロックの入力パラメータの値をＩＯメモリ１５から読み出して、ＩＯメモリ１５のファンクションブロック実行領域にその値をコピーする（ＳＴ３）。このようにコピーするのは、ファンクションブロック用のＩＮデータおよびＯＵＴデータの記憶領域を別途設けているからである。次いで、ＦＢボディ処理として、命令実行エンジン１６は、ファンクションブロック実行領域を参照してボディに記述されたラダープログラム（図２（ｂ）等）を実行し、その実行結果をＩＯメモリ１５のファンクションブロック実行領域へ格納する（ＳＴ４）。その後、インスタンス後処理として、ＩＯメモリ１５のファンクションブロック実行領域に記憶されている実行結果の値を出力パラメータ領域にコピーし、ファンクションブロック部分の処理を終了する（ＳＴ５）。次いで、命令実行エンジン１６は、ユーザメモリＵＭ１４を再び参照し、ファンクションブロックに続くユーザプログラムのメインルーチンの続きを実行する（ＳＴ６）。なお、係るファンクションブロックの実行処理手順自体は、従来公知のものであるので、その詳細な説明を省略する。

【００２７】

ここで、本発明では、通常のユーザメモリ（表ユーザメモリ）１４に加え、テンポラリユーザメモリ（裏ユーザメモリ）１９を設けた。ユーザメモリ１４は、通常動作時のユーザプログラムを格納する領域であり、そして、命令実行エンジン１６はこのユーザメモリ１４からユーザプログラムを呼び出して実行する。また、プログラミングツール２０からのアクセス許可領域は、ユーザメモリ１４であり、もう一方のテンポラリユーザメモリ１９に対してはアクセスできないようになっている。これは、ユーザプログラムを参照する場所とツール２０がアクセスする場所とを区別できるようにするためである。詳しくは図６，７などで後述するが、つまり、命令実行エンジン１６がユーザプログラムを実行する際に、その実行中にツール２０により編集された直後のプログラムを参照しないようにするためである。もし命令実行エンジン１６が実行中に編集プログラムを随時参照すると、プログラム内容が実行途中で変更されることとなり、動作が不安定になるからである。

【００２８】

また、テンポラリユーザメモリ１９は、ユーザメモリ１４に格納されたユーザプログラムと同一のプログラムが格納されている。オンラインエディット時には、命令実行エンジ

ン１６はユーザメモリ１４を参照して実行せずに、代わりにテンポラリユーザメモリ１９に格納されたユーザプログラムを呼び出して実行する。この切り替えにより、オンラインエディット中も、オンラインエディット前のユーザプログラムと同一内容のユーザプログラムに基づいた演算処理を安定して続けることができる。

【００２９】

なお、ユーザメモリ１４とテンポラリユーザメモリ１９は、物理的に異なるメモリ（記憶素子）を用いても良いし、同一のメモリを用いてメモリ割付によって２つのメモリ領域を設ける形態で実現しても良い。

【００３０】

さて、オンラインエディット時のツール２０の操作・動作を図４に沿って説明する。まず、編集対象のユーザプログラムをＣＰＵユニット１０のユーザメモリ１４からアップロードをする（ＳＴ１１）。このときＣＰＵユニット１０側はオンラインエディット中の処理をしている。詳しくは図６で説明するが、このときＣＰＵユニット１０が実行対象としているユーザメモリはテンポラリユーザメモリ１９のほうなので、アップロードする対象メモリはもう一方のユーザメモリ１４であるほうが望ましい。これは、ＣＰＵユニット１０が参照しているメモリとプログラミングツール２０が参照しているメモリとを区別したほうが制御にとって安全だからである。ユーザメモリ１４のプログラムが格納されている全ての領域についてアップロードが完了すると、ツール２０にてオペレータがユーザプログラムを編集する。この編集というのは、ユーザプログラムの追加、削除、変更などを行うことである（ＳＴ１２）。ユーザプログラムの編集が終われば、ツール２０からＣＰＵユニット１０のユーザメモリ１４へ編集後のユーザプログラムをダウンロードする（ＳＴ１４：図５（ａ）参照）。ツール２０からのダウンロードを受けるＣＰＵユニット１０にとっては、このダウンロードはサイクリック処理中の周辺処理期間にて行われることとなる。ダウンロードするプログラムの容量が多い場合には、複数回サイクルの周辺処理にまたがって分割してダウンロードを行っても良い。

【００３１】

ツール２０は、ダウンロードが正常に完了するどうかを確認する（ＳＴ１５）。その確認は、ＰＬＣ側でダウンロード完了かどうかを自己判断し、完了したならばツール２０に対してその旨を返信し、その返信をツール２が受信することでもよい。ＰＬＣ側の自己診断は、例えばＭＰＵ１２がユーザメモリ１４に格納されたコードデータのサム値チェックなどにより正常にダウンロード完了できたか否かを判断する。もし正常に完了できていない場合には、ツール２０が、ＭＰＵ１２からダウンロード不完全である旨の通知をうけ、再度ダウンロードを実行するようになっている。

【００３２】

この後、ユーザメモリ１４にダウンロードしたユーザプログラムをテンポラリユーザメモリ１９にコピーして記憶保持する。ＰＬＣ側でダウンロードが完了したか否かを自己診断する場合には、このユーザプログラムコピー格納処理は、ＭＰＵ１２がダウンロードの正常完了を判断することによって開始される。この場合のコピー格納処理は、ＣＰＵユニット１０におけるサイクリック処理中のユーザプログラム実行期間以外、例えば周辺処理期間で行われる。また変形例として、ツール２０が編集プログラムをＣＰＵユニット１０へダウンロードする際に、ツール自身がＭＰＵ１２に対して、ユーザメモリ１４への格納処理命令とテンポラリユーザメモリ１９へのコピー格納処理命令とを発信するようにしてもよい。（図５（ｂ）参照）

【００３３】

次に、図６に示すフローチャートに沿って、ＣＰＵユニット１０側のオンラインエディット中の処理について説明する。ここではファンクションブロックの追加を、ＣＰＵユニットのサイクリック処理動作中に実行する場合を例とする（具体的な編集内容は図９を説明する際に後述する）。まず、ツール２０がオンラインエディット処理をするべく、ユーザプログラムのアップロードを行おうとする前に、ＣＰＵユニット１０の命令実行エンジン１６におけるプログラム実行対象をテンポラリユーザメモリ１９に切り替える（ＳＴ２

１）。この切り替えは、命令実行エンジン１６の実行対象をユーザメモリ１４にするかテンポラリメモリ１９にするかを示す特定フラグを書き換えることにより、実現できる。また変形例として、プログラミングツール２０によってオンラインエディットする際に、プログラミングツール２０から自動的に、ＣＰＵユニット１０へ命令実行エンジン１６の実行対象をテンポラリユーザメモリ１９に切り替える命令を出力するようにしてもよい。いずれかの手法によってＳＴ２１の切り替えを行うことにより、命令実行エンジン１６に対しては、テンポラリユーザメモリ１９に格納されたユーザプログラム（命令オブジェクト）が提供され、実際の制御が継続される。ユーザメモリ１４の内容とテンポラリユーザメモリ１９の内容が一致しているため、命令実行エンジン１６は今まで処理していたのと同一内容のユーザプログラムを継続して実行することができる。

【００３４】

次に、プログラミングツール２０によりユーザメモリ１４に格納されている編集前のユーザプログラムをすべてアップロードされる（なお変形例として、編集対象となる部分だけをアップロードすることも考えられる）。このアップロード処理はＣＰＵユニットにおけるサイクリック処理中の周辺処理期間に行われる。ユーザプログラムの容量が多い場合には、複数サイクルの周辺処理期間に分割してツール２０に対して送信してもよい。ツール２０側では、このアップロード送信が終了してから実際のユーザプログラム編集処理に入る（前述の図４のＳＴ１２）。次に、ＭＰＵ１２は、ユーザメモリ１４の領域のうち、追加するファンクションブロックのボディプログラム領域を確保する（ＳＴ２２：図７（ａ）参照）。ここでユーザメモリ１４のメモリマップについて図８を用いて説明する。メモリマップは、図８に示すように、先頭にプログラムの基本情報（サム値など）を記録する領域があり、次に、使用されているファンクションブロックのボディ部のプログラム情報を記憶するＦＢレコード部と、ユーザプログラムおよび付随するファンクションブロックのインスタンスを格納するプログラムレコード部がある。ＳＴ２２では、ＦＢレコード部における未設定の領域を、今回追加するファンクションブロックの記憶領域として特定する。

【００３５】

図９に示すように、所定位置にＦＢ　Ｚ＿４を追加する編集をユーザプログラムに対して行う際には、ＭＰＵ１２が、新規に追加するＦＢ　Ｚ＿４のためにＦＢレコード部の書き込む領域を特定して確保する。図８のようにＦＢレコード部は、予めファンクションブロックが数多く書き込めるように区分けされていて、その区分け領域のアドレスが小さい領域から順にファンクションブロックを格納してある。よって、新規なＦＢ　Ｚ＿４のための領域確保をすることは、まだＦＢが格納されていない領域のうちアドレス値が小さい領域を確保することとなる。このようにＦＢ記憶領域を特定した後、ツール２０側にてその確保領域に格納しようとするファンクションブロックを編集する。同時に、ユーザプログラム中における追加対象のファンクションブロックを実行する該当位置（プログラムレコード部のプログラムタスクの該当記憶領域の位置）に、その追加ファンクションブロックのために確保したＦＢレコード部の該当記憶領域の先頭アドレスを呼び出すプログラムも記述する。この呼び出しプログラムについては、ユーザが都度に手入力してもよいし、ツール機能によって自動化してもよい。この場合、ＦＢ記憶領域を確保したときにツール２０はその先頭アドレスを把握しているので、その情報を用いて呼出プログラムを自動作成することができる。

【００３６】

プログラミングツール２０において、ユーザプログラムの編集（ファンクションブロックの追加、インスタンスの設定を含む）が終わると、ユーザメモリ１４の確保したＦＢ記憶領域に対して、追加したＦＢボディブロックがダウンロードされてくる。ＭＰＵ１２は、受信したＦＢボディブロックをユーザメモリ１４の確保しておいた領域に格納する。次に、ユーザメモリ１４のプログラムレコード部に対して、ユーザプログラム中に追加ファンクションブロック記憶領域（今回、特定・確保したＦＢ記憶領域）を呼び出すための呼出プログラムが、ツール２０からダウンロードされてくる。ＭＰＵ１２は同じように確保

した領域に格納する（ＳＴ２３、ＳＴ２４：図７（ｂ）参照）。なお、ここでは編集部分だけをダウンロードされて格納する処理を説明したが、変形例として、ツール２０から編集したユーザプログラムを全部（編集箇所、編集していない箇所とも全部）についてユーザメモリ１４にダウンロードしてもよい。なお、ツール２０から編集プログラムをユーザメモリ１４へダウンロードする処理は、ＣＰＵユニット１０にとっては、ＭＰＵ１２が通信Ｉ／Ｆ１８を介してなされる処理であり、サイクリック処理中の周辺処理期間に行われる処理となる。

【００３７】

次に、ＭＰＵ１２にて、ユーザメモリ１４のプログラム基本情報を参照し、サム値チェックなどにより正常にダウンロードしたか否かを判断する（ＳＴ２５）。変形例として、この処理をＭＰＵ１２に代わって、ツール２０が行っても良い。その場合は、ツール２０がユーザメモリ１４のプログラム基本情報を読み出して、サム値チェックの処理をすることとなる。

【００３８】

ＭＰＵ１２が、ダウンロード処理を正常に完了したと判断すると、ユーザメモリ１４に記憶されたユーザプログラムが、図９（ａ）の状態から図９（ｂ）に示すように変更されたことになる。そこで、ＭＰＵ１２は、命令実行エンジン１６におけるプログラム実行対象をテンポラリユーザメモリ１９からユーザメモリ１４へ戻す（ＳＴ２６：図７（ｃ）参照））。この切り替えに伴い、命令実行エンジン１６は、ユーザメモリ１４に格納された編集後のユーザプログラムを実行することになる。なおこの切り替え処理は、ＣＰＵユニット１０側で正常にダウンロード完了判断したことによって行われるが、ツール２０側でダウンロード完了判断をするようにした場合にはツール２０から切り替え命令をＣＰＵユニット１０が受信し、またはツール２０から前述の切り替えフラグが更新されることによって行うようにしてもよい。またこの切り替え処理は、ＣＰＵユニット１０におけるサイクリック処理のユーザプログラム実行期間以外の期間で行うこととなる。例えば周辺処理期間に行う。

【００３９】

次いで、ＭＰＵ１２が、ユーザメモリ１４の内容をテンポラリユーザメモリ１９にコピーする処理を行う。（ＳＴ２７：図７（ｄ）参照）。実際には、一度にユーザメモリの全体をコピーすることはできないので、ＰＬＣのサイクリックな処理中の周辺処理時に少しずつコピーすることになる。このコピー処理に伴ない、編集後のユーザプログラムがテンポラリユーザメモリ１９にも記憶されることとなる。よって、次回のオンラインエディット時にも、命令実行エンジン１６の実行対象をユーザメモリ１４からテンポラリユーザメモリ１９に切り替えることで、ツール２０がアクセスするユーザメモリ１４とは区別できるので、命令実行エンジン１６はその時点のユーザプログラムを安定してそのまま実行継続できる。

【００４０】

なお、命令実行エンジンがより広範囲の処理に対応できるものであれば、上記のダウンロードされてきたユーザプログラムを受信してユーザメモリ１４へ格納する処理や、ユーザメモリ１４とテンポラリユーザメモリ１９との参照切替え処理や、編集後のユーザプログラムをユーザメモリ１４からテンポラリメモリ１９へのコピー処理などを、ＭＰＵ１２からの指示を受けることによって命令実行エンジン１６が行うようにしてもよい。また、ＭＰＵ１２の機能と命令実行エンジン１６の機能をひとつのプロセッサで実現するようにした場合には、これらの処理はそのプロセッサが行うこととなる。

【００４１】

また、上記した実施の形態では、編集処理の一態様であるファンクションブロックの追加について説明したが、ファンクションブロックの削除編集も同様に行える。もちろん、ファンクションブロック以外に、メインのユーザプログラムの通常のラダー演算部分の複数箇所を編集する場合にも同様に行える。そして編集対象となるプログラム数が１個でも適用はできる。つまり、いずれの場合にも編集処理をするに先立ち、プログラム実行をテ

ンポラリユーザメモリ１９側に移す。これにより、ユーザメモリ１４に対してオンラインエディットによる編集処理（アップロード、ダウンロード処理）をしても、命令実行エンジン１６は、ツール２０によってアクセスされるメモリ領域とは別のメモリ領域を参照するので、ユーザプログラム実行自体には影響を与えなくなる。

【００４２】

なお、ユーザメモリとテンポラリユーザメモリの扱いについて、上記の実施形態ではオンラインエディット処理をするときに命令実行エンジンの実行対象をＵＭからＴ－ＵＭに切り替えていたが、別の実現方法があるので変形例について説明しておく。その変形例では、オンラインエディット処理をはじめるときに、上記の実施形態（図６のＳＴ２１）のように命令実行エンジンの参照対象のメモリを切り替えるのではなく、予めメモリの扱いの定義を変えておくことで実現できるようにした。

【００４３】

オンラインエディット処理をするとき、命令実行エンジンが参照しているメモリは参照用メモリ（テンポラリメモリＴ－ＵＭ）と定義されていて、同一のユーザプログラムが格納されているもう一方のメモリ（今時点で命令実行エンジンが参照していないほうのメモリ）をエディット用メモリ（ユーザメモリＵＭ）と定義されている（図１０（ａ）参照）。そしてユーザプログラムを編集した後、エディット用メモリＵＭにその編集プログラムをダウンロードして格納する（図１０（ｂ）参照）。ダウンロードが完了した後、命令実行エンジンが参照するメモリを参照用メモリＴ－ＵＭからエディット用メモリＵＭに切り替える。この切り替えによって、命令実行エンジンは編集後プログラムを参照して実行するようになる。またこの時点ではＴ－ＵＭのほうのメモリには編集前のプログラムが格納されている（図１０（ｃ）参照）。ここまでは図７の説明と同じである。

【００４４】

次に、各メモリの扱いの定義を変える。つまり、現時点でエディット用メモリＵＭであるほうのメモリを参照用メモリＴ－ＵＭへ定義変更するとともに、参照用メモリＴ－ＵＭであるほうのメモリをエディット用メモリＵＭへ定義変更する（図１０（ｄ）参照）。この定義変更によって、ツールは定義変更後の参照用メモリＴ－ＵＭにアクセスできなくなる。そして、定義変更後のエディット用メモリＵＭには編集後のプログラムがまだ格納されていないので、参照用メモリＴ－ＵＭに格納されている編集後プログラムをエディット用メモリＵＭにコピーする（図１０（ｅ）参照）。

【００４５】

次回、オンラインエディット処理をするときには、ツールは、定義変更後のエディット用メモリＵＭに対して、エディット処理を行う。つまり、エディット用メモリＵＭに対し、そのＦＢボディ領域を確保することや、編集後のプログラムをダウンロードする（図１０（ｆ）参照）。ダウンロード完了すれば、前述と同様に命令実行エンジンの参照対象を切り替え、メモリの定義を入れ替え、編集後のプログラムをコピー格納する。このようにすることにより、ツール２０がアクセスするメモリ（ユーザメモリＵＭ）と、命令実行エンジン１６が参照するメモリ（テンポラリユーザメモリＴ－ＵＭ）とは区別できるので、命令実行エンジン１６はユーザプログラムを安定してそのまま実行継続できる。

【００４６】

なお、プログラミングツール２０からアクセスできるメモリをユーザメモリ１４だけに制限した例を説明したが、ツールがメモリにアクセスする場合の変形例が考えられる。プログラミングツール２０がユーザプログラムをアップロードする際には、ユーザメモリとテンポラリユーザメモリとには同一内容のユーザプログラムが格納されているので、命令実行エンジン１６が参照対象としているほうのメモリでもよい。これはアップロードつまり読出し処理をすることによってメモリの内容が変化するわけではないからである。結局、アップロード時には、プログラミングツール２０から、命令実行エンジン１６が参照対象しているほうのメモリへも、参照対象でないほうのメモリへもアクセスしてもよい。いっぽう、編集後のユーザプログラムをダウンロードする際には、メモリ内容を書き換えることとなるので、命令実行エンジン１６が参照対象していないほうのメモリに対してダウ

ンロードする必要がある。

【００４７】

プログラマブルコントローラの変形例について説明しておく。上述の実施形態では、複数のユニットから構成されるＰＬＣを示した。このＰＬＣは電源ユニット，ＣＰＵユニット，入力ユニット，出力ユニット，通信ユニットなどの各種のユニットを組み合わせて構成されるものであった。この発明はそのようなＰＬＣに限らず、例えば、各ユニットの機能を1つのユニットにより一体化したＰＬＣであってもよい。これはマイクロＰＬＣと呼ばれる１ユニット型プログラマブルコントローラである。このマイクロＰＬＣは通信ポートを備えているので、ネットワークを介してプログラミングツールを接続することができる。このようなプログラマブルコントローラであってもこの発明を適用することができる。

【図面の簡単な説明】

【００４８】

【図１】本発明に係るＣＰＵユニットの一実施の形態を示すブロック図である。

【図２】ファンクションブロックを説明する図である。

【図３】ファンクションブロックを含むラダープログラムの実行処理を示すフローチャートである。

【図４】ＭＰＵの機能（ダウンロード）を示すフローチャートである。

【図５】ＭＰＵのダウンロードの作用を説明する図である。

【図６】ＭＰＵの機能（ファンクションブロックの追加）を示すフローチャートである。

【図７】ＭＰＵの機能（ファンクションブロックの追加）の作用を説明する図である。

【図８】ユーザメモリのメモリマップの一例を示す図である。

【図９】ＭＰＵの機能（ファンクションブロックの追加）の作用を説明する図である。

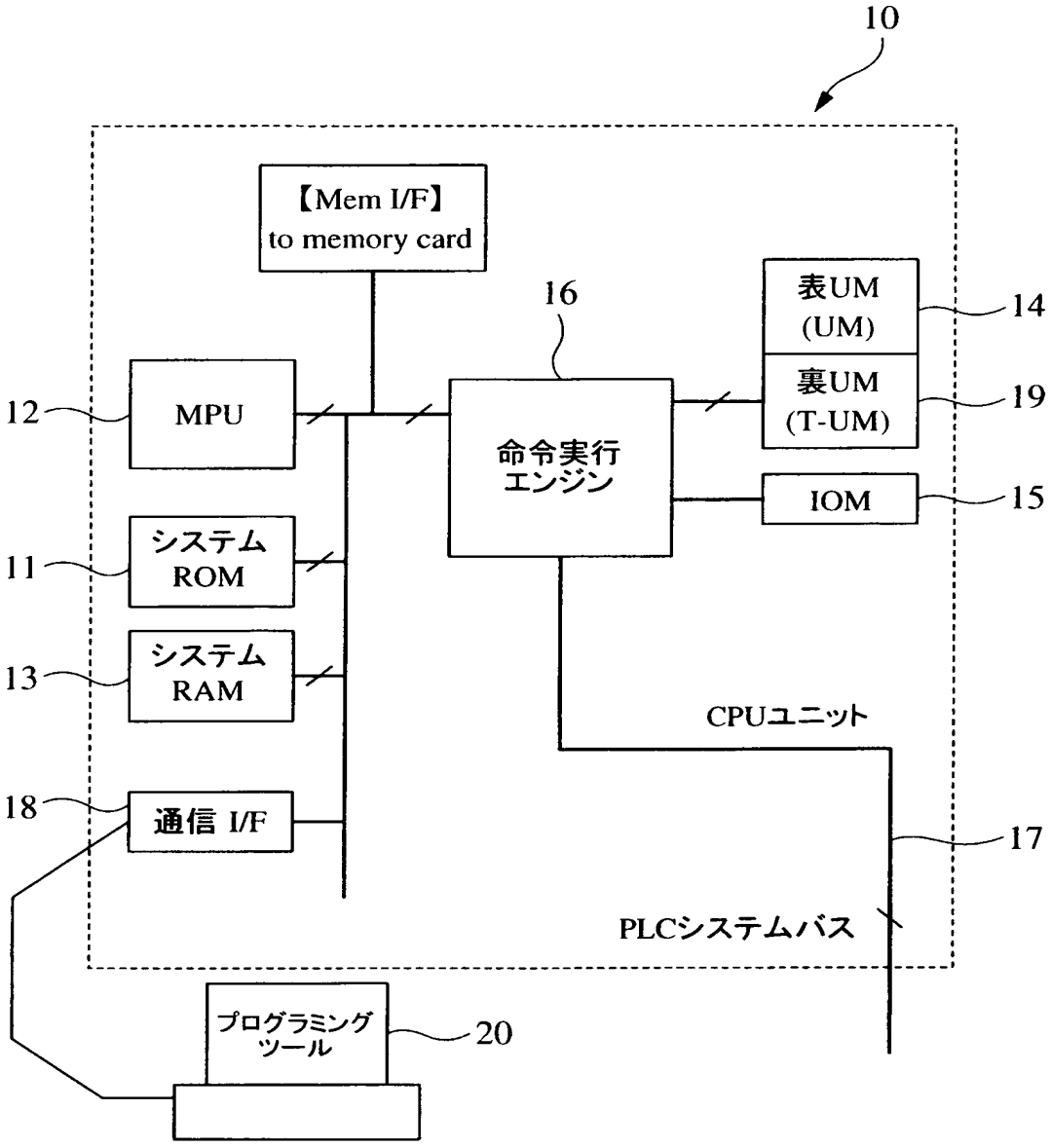
【図１０】ＭＰＵの機能（ファンクションブロックの追加）の作用における変形例を説明する図である。

【符号の説明】

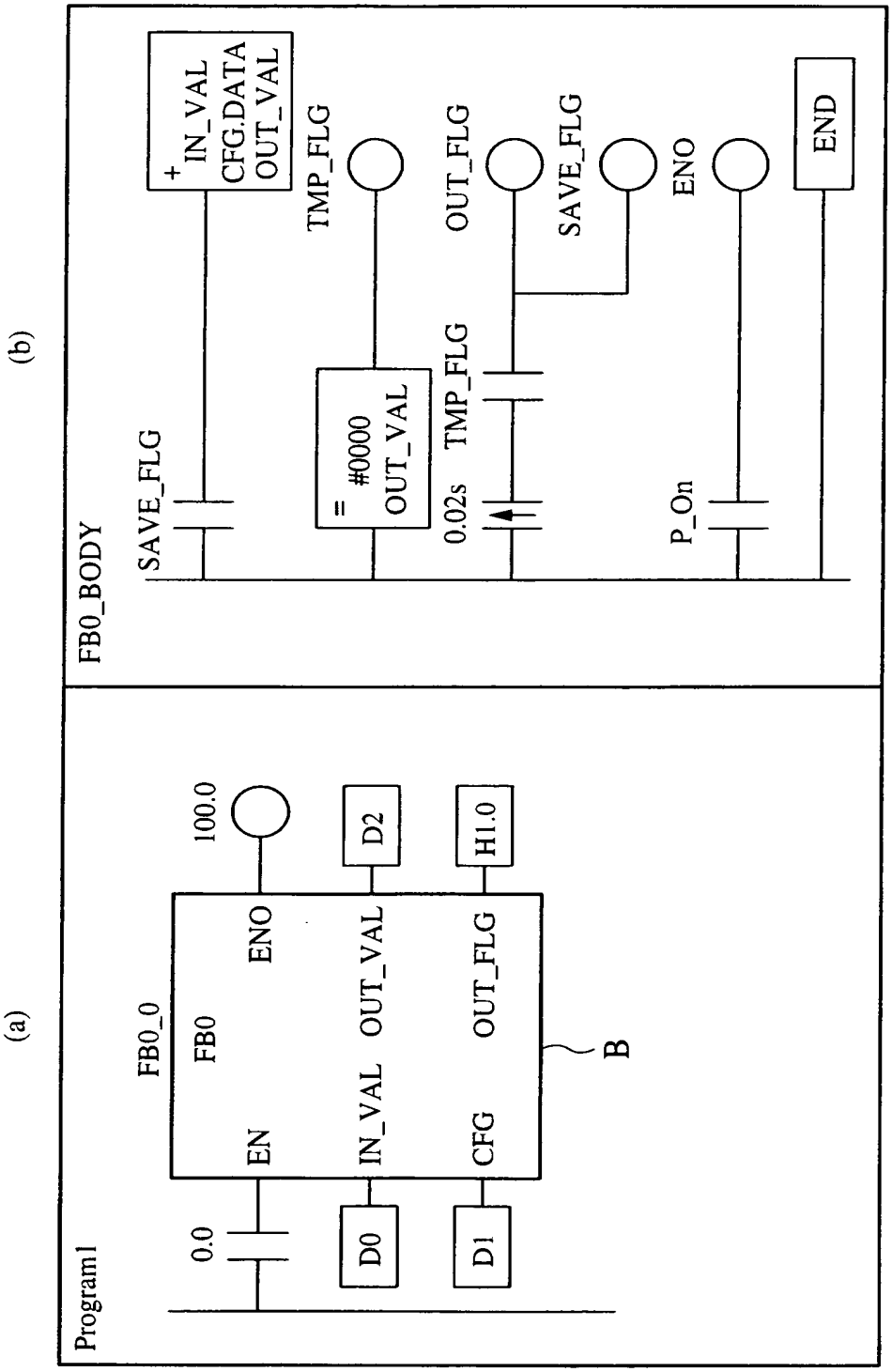
【００４９】

１０　ＣＰＵユニット
１１　システムＲＯＭ
１２　ＭＰＵ
１３　システムＲＡＭ
１４　ユーザメモリ
１５　ＩＯメモリ
１６　命令実行エンジン
１７　システムバス
１８　通信インタフェース
１９　テンポラリユーザメモリ
２０　ツール

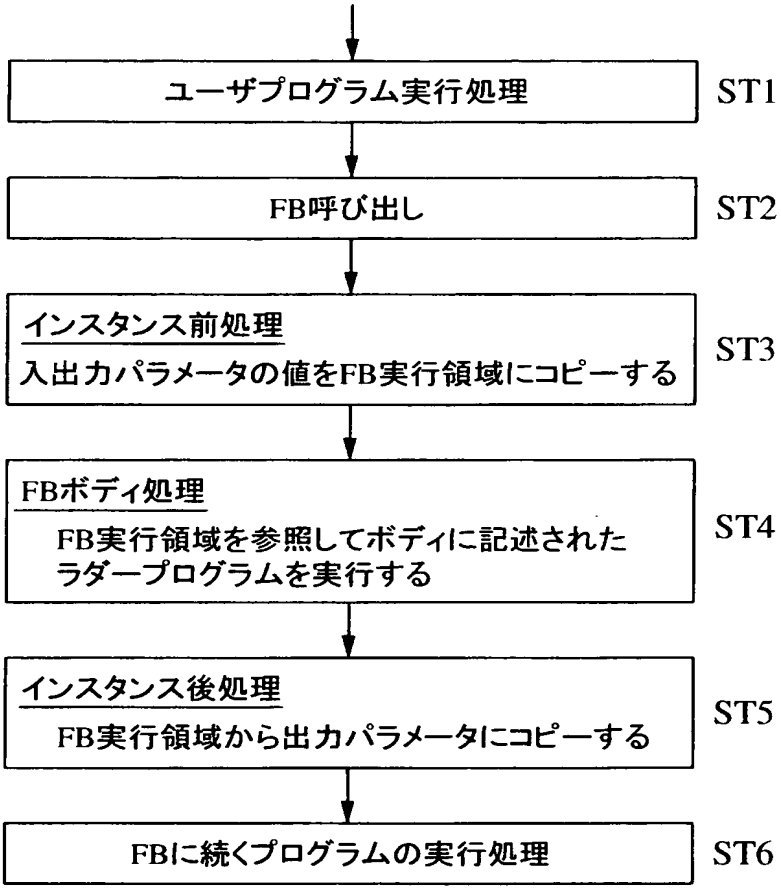
【書類名】図面
【図１】


10
【Mem I/F】
to memory card
16
表UM
(UM)
14
12
MPU
裏UM
(T-UM)
19
命令実行
エンジン
IOM
15
11
システム
ROM
13
システム
RAM
CPUユニット
18
通信 I/F
17
PLCシステムバス
プログラミング
ツール
20

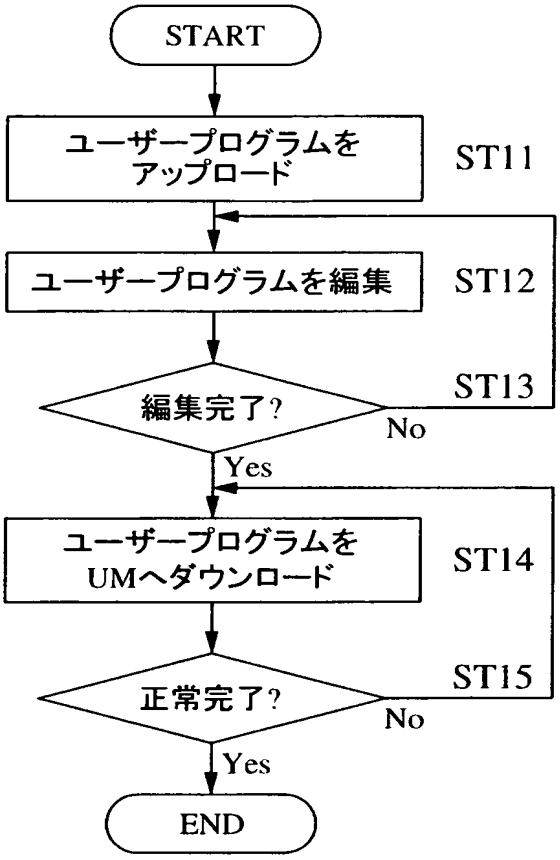
【図２】


(a)
Program1
0.0
EN
FB0_0
FB0
ENO
100.0
IN_VAL
OUT_VAL
OUT_FLG
CFG
D0
D1
D2
H1.0
B
(b)
FB0_BODY
SAVE_FLG
+
IN_VAL
CFG.DATA
OUT_VAL
TMP_FLG
=
#0000
OUT_VAL
0.02s
TMP_FLG
OUT_FLG
SAVE_FLG
P_On
ENO
END

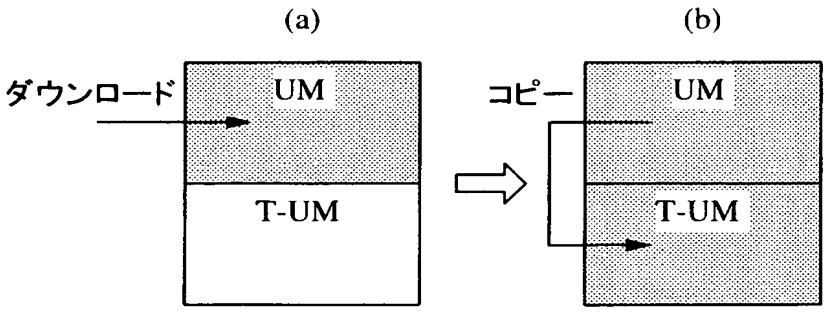
【図３】


ユーザプログラム実行処理
ST1
FB呼び出し
ST2
インスタンス前処理
入出力パラメータの値をFB実行領域にコピーする
ST3
FBボディ処理
FB実行領域を参照してボディに記述された
ラダープログラムを実行する
ST4
インスタンス後処理
FB実行領域から出力パラメータにコピーする
ST5
FBに続くプログラムの実行処理
ST6

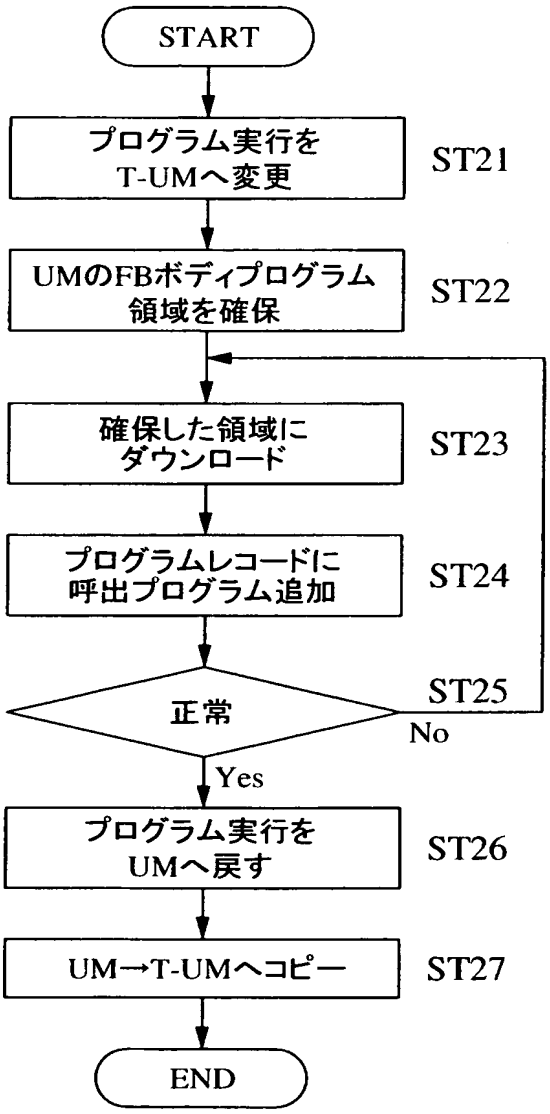
【図４】


START
ユーザープログラムを
アップロード
ST11
ユーザープログラムを編集
ST12
編集完了?
ST13
No
Yes
ユーザープログラムを
UMへダウンロード
ST14
正常完了?
ST15
No
Yes
END

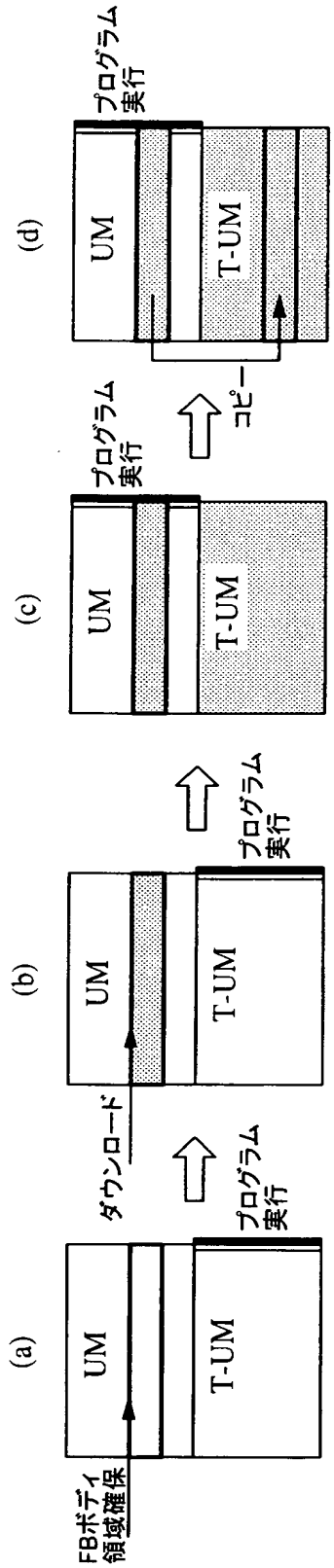
【図５】


(a)
(b)
ダウンロード
UM
T-UM
コピー
UM
T-UM

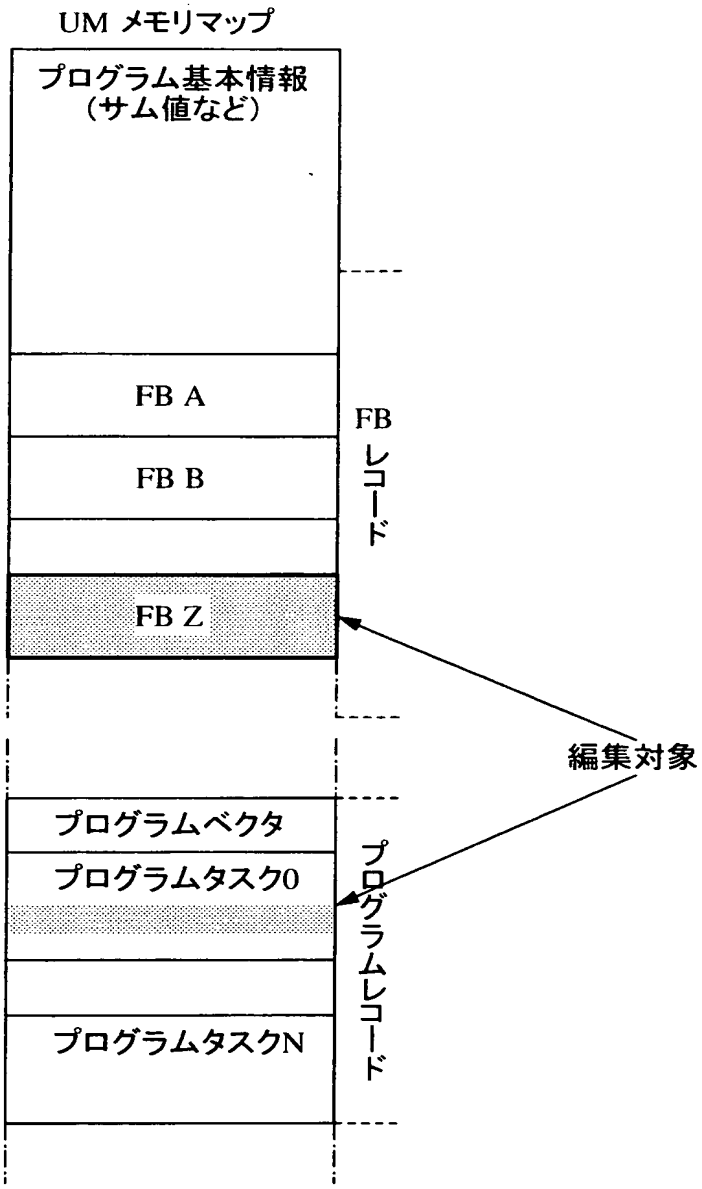
【図６】


START
プログラム実行を
T-UMへ変更
ST21
UMのFBボディプログラム
領域を確保
ST22
確保した領域に
ダウンロード
ST23
プログラムレコードに
呼出プログラム追加
ST24
正常
ST25
No
Yes
プログラム実行を
UMへ戻す
ST26
UM→T-UMへコピー
ST27
END

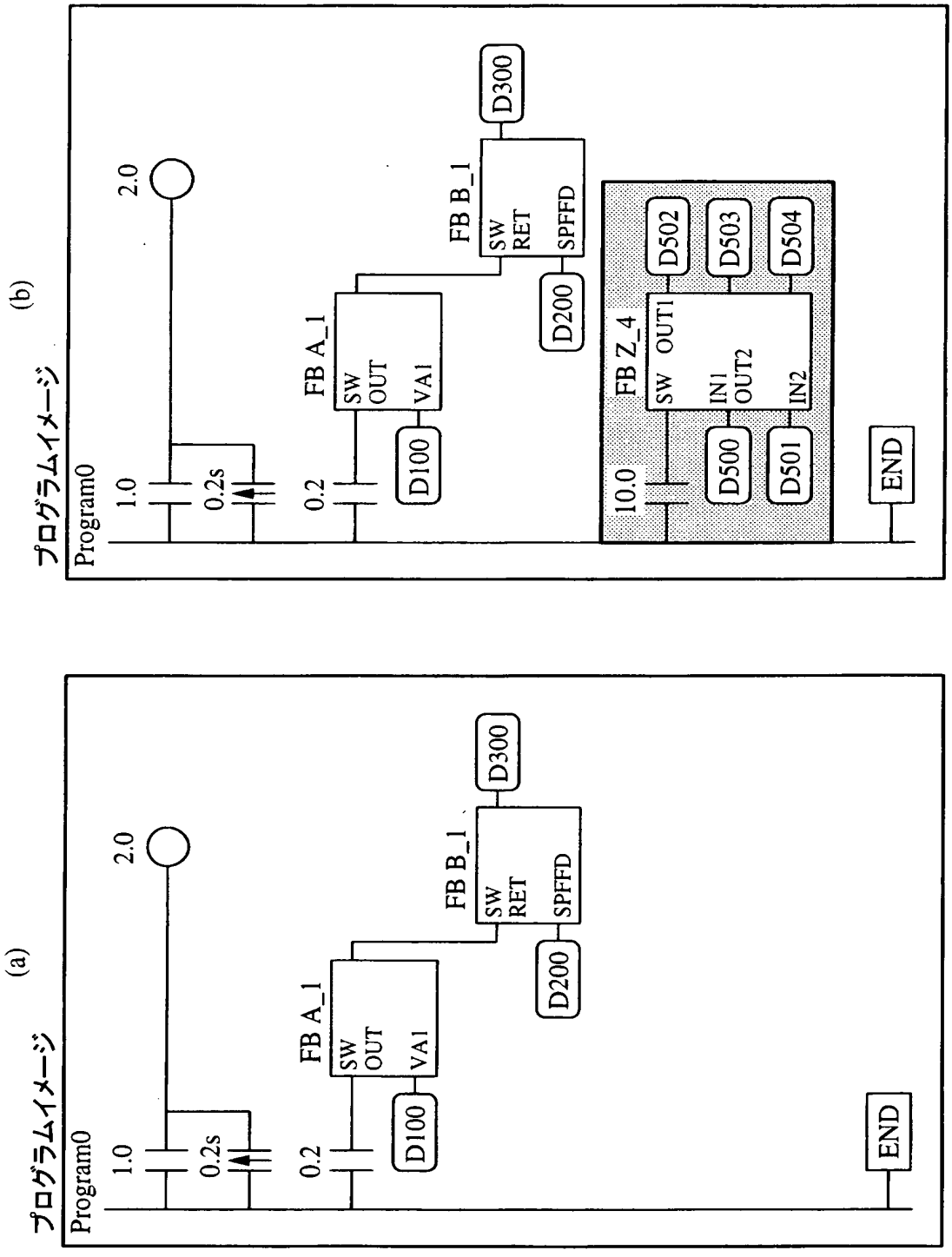
【図７】


(a)
UM
T-UM
FBボディ
領域確保
ダウンロード
プログラム
実行
(b)
UM
T-UM
プログラム
実行
(c)
UM
T-UM
プログラム
実行
コピー
(d)
UM
T-UM
プログラム
実行

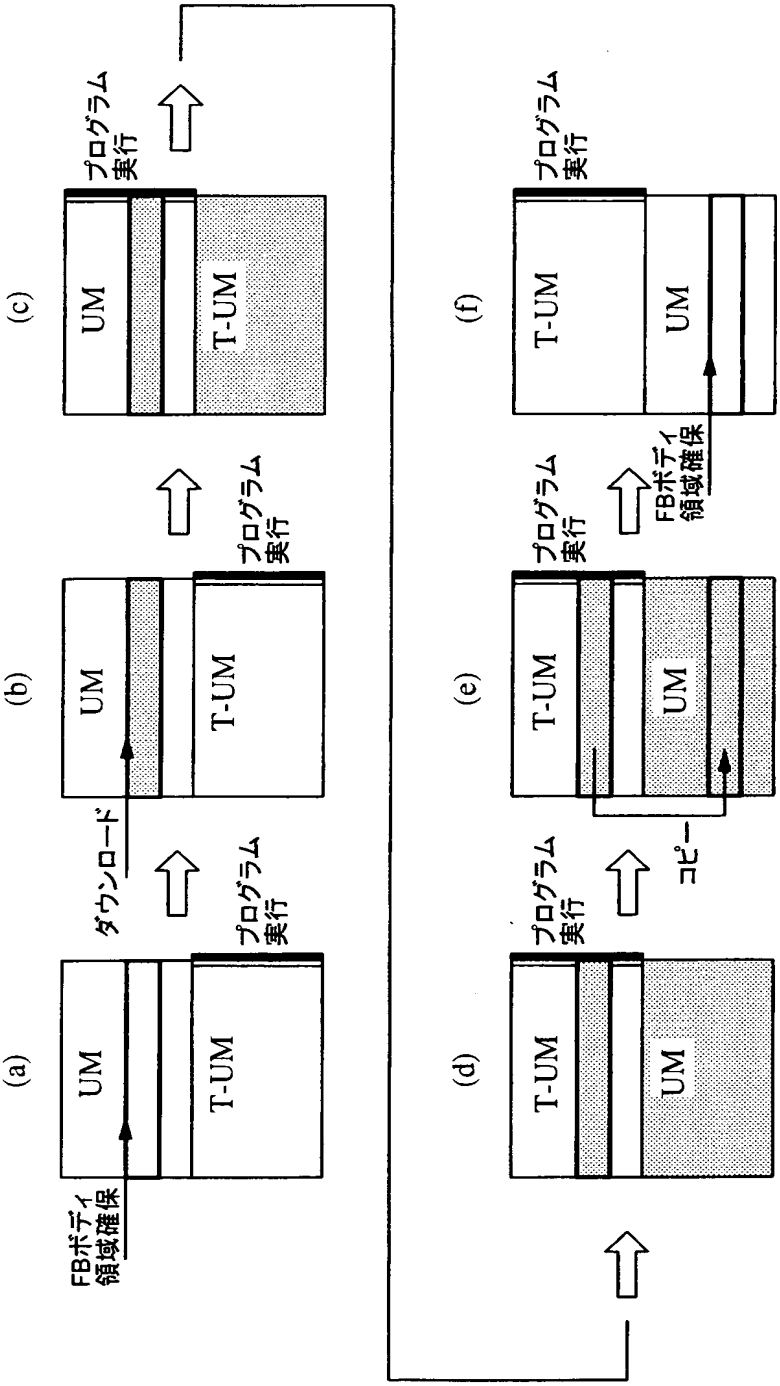
【図８】


UM メモリマップ
プログラム基本情報
（サム値など）
FB A
FB B
FB Z
FB
レコード
編集対象
プログラムベクタ
プログラムタスク0
プログラムタスクN
プログラム
レコード

【図９】


(a)
プログラムイメージ
Program0
1.0
2.0
0.2s
0.2
FB A_1
SW
OUT
VA1
D100
FB B_1
SW
RET
SPFFD
D300
D200
END
(b)
プログラムイメージ
Program0
1.0
2.0
0.2s
0.2
FB A_1
SW
OUT
VA1
D100
FB B_1
SW
RET
SPFFD
D300
D200
FB Z_4
SW
OUT1
IN1
OUT2
IN2
D502
D503
D504
10.0
D500
D501
END

【図１０】


(a)
UM
T-UM
FBボディ領域確保
プログラム実行
(b)
UM
T-UM
ダウンロード
プログラム実行
(c)
UM
T-UM
プログラム実行
(d)
T-UM
UM
プログラム実行
(e)
T-UM
UM
コピー
プログラム実行
(f)
T-UM
UM
FBボディ領域確保
プログラム実行

【書類名】要約書
【要約】
【課題】　オンラインエディットによるプログラムの修正が容易に行うことができ、かつ、編集対象（ファンクションブロックか否か）並びに個数に関係なく編集処理が行えるＣＰＵユニットを提供すること
【解決手段】　ユーザメモリ１４に格納されたユーザプログラムを演算実行するプログラマブルコントローラにおけるＣＰＵユニット１０である。そして、ユーザプログラムと同一のユーザプログラムを格納するテンポラリユーザメモリ１９を設け、ユーザメモリに対する編集処理を行う際には、テンポラリユーザメモリに格納されたユーザプログラムが命令実行エンジン１６に呼び出され、演算実行される。編集後は、ユーザメモリに格納された編集済みユーザプログラムに基づき実行され、その編集済みユーザプログラムは、テンポラリユーザメモリにコピーされる。
【選択図】　図１

特願２００４－０５９８６５

出　願　人　履　歴　情　報

識別番号　［０００００２９４５］

１．変更年月日　２０００年　８月１１日
［変更理由］　住所変更
住　所　京都市下京区塩小路通堀川東入南不動堂町８０１番地
氏　名　オムロン株式会社